Too Many Eyes / Potion System

System Design Document

# Changes

## V 1.0

Editor: Perrin Peterson

10/11/2022

* Document Created, and initial version

## V 2.0

Editor: Perrin Peterson  
12/08/2022

* Major overhaul and rework.

# Introduction

This document details the design and purpose of a locomotion system designed for Too Many Eyes. Designed for the Blight Brew Game, the document will cover;

* Usage for the designers.
* UML for the Programmers.

# Design Goals

This system is to give the player a commodity to craft and sell in the shop, as well as tools to complete puzzles and assist in exploration.

The selling and crafting are explored in other documents MDD’s, but the potions as tools are explored here. The potions currently can be consumed and thrown, applying the effects of the potion on either the player, or the character/object the potion hits. Depending on the effects, this can lead to the completion of puzzles and defeating of enemies.

# Behaviour

This system focuses on making a component based system for potions that allows for maximum freedom when designing the potions, while requiring minimal updates for the programmers. For example;

* Components
  + Consumable Component
  + Stamina Effect Component
* Potions that can be made:
  + Potion of Stamina Restore (Adds to the players current stamina)
  + Potion of Max stamina (Adds to the players max stamina)
  + Draining potion of Max Stamina (Subtracts from the players Current stamina, but increases max stamina)
  + Potion of Stamina Regen (Adds to the stamina regen of the player)
  + Many more…
  + And all of these can have “Variants” like Long Lasting, or Cursed.

This gives the programmers only 1 or 2 components they need to make, so that the designers can make an increasingly large number of potions. Components can be mixed as well, making a throwable AND consumable potion, or a stamina AND health restore is an option for the designers.

In version 2.0 of this system, the overhaul gave us some major freedom bonuses. Using a custom made component system, we can apply specific effects to specific components. For example, we can have a potion that heals when drank, and harms when thrown (See example below). This gives even more options with only a few effects and components.

Graphical user interface, text, application, chat or text message

Description automatically generated

# High Level Design

Graphical user interface

Description automatically generated with medium confidence

* Player system - the system is our core system and is what the player interacts with to use all our other systems.
* Locomotion system - an advanced movement system we’re using to give us more control over the players movement. This gives us variables we can control, such as stamina, and additional functionality, such as climbing and vaulting.
* Potion system – a system meant to be easy to use for designers. The system allows for the designers as much freedom as possible, while giving the coders a minimal amount of updating to do. This is the main system the player will use to complete puzzles and generate income.
* Inventory system – a system meant to be easy to use for designers. The system allows for the designers as much freedom as possible, while giving the coders a minimal amount of updating to do. The system works as a container to hold items that the player collects. This includes the players backpack, as well as storage containers around the world.
* Resource system – a system meant to be the first of the steps for the player to generate income. This system gives the player ways to harvest materials and shows off behavioural logic for when the system is used.
* Crafting System – the system in between the Resource system, and the Shop/Potion systems, in terms of actual gameplay. This systems job is to refine the resources the player gathers into other items or potions for use in one of the other systems.
* Shop system (Not yet implemented) – a system to generate income, and allow for the player to improve. The system interfaces with AI, the player, and Storage objects, to give the player a way of selling items to NPC’s. This can have numerous effects on the AI and is the primary source of income, a necessary resource for improving the players arsenal, and serves to break the monotony of just grinding for resources.
* AI System - The system dealing with the NPC’s in the world, including fauna. The system interfaces with the shop system, allowing for NPC’s to be customers, as well as gives the AI their logic for movement, interaction, and anything else.

# Mid Level View

Graphical user interface

Description automatically generated

* IBBInterface is inherited by any world objects that we want to have “Focus Text” which is displayed when the player is close and looking at them.
* ABBBaseItem is the basis of all the Actors that can be picked up and stored in the player’s inventory.
* ABBPotionBase is the class that AL potions are blueprinted from, and then customized in the editor. It has a projectile component, so that the throwable effects can be applied.
* UBBPotionComponent is the Parent of all the components that can be attached to potions. The children of this class will contain logic that the potion can then execute. It contains an internal list of effects that can be added in the editor, and those effects can be executed by the OnUse function, defined by the children.
  + UBBThrowableComponent makes use of the Projectile component and allows the potion to be thrown.
  + UBBConsumableComponent applies effects directly to the player.
* UBBEffect is the parent of all the effects that a component can apply. Using a parent class gives us the ability to use inheritance and allows many kinds of effects to be called by one component.
  + UBBStaminaEffect effects the stamina component of whatever Character is passed into it’s OnUse function. It can do a variety of effects including modifying max and adjusting current stamina.

# Logical View

Text

Description automatically generatedText

Description automatically generated Text

Description automatically generated

Table

Description automatically generated A picture containing table

Description automatically generated Table

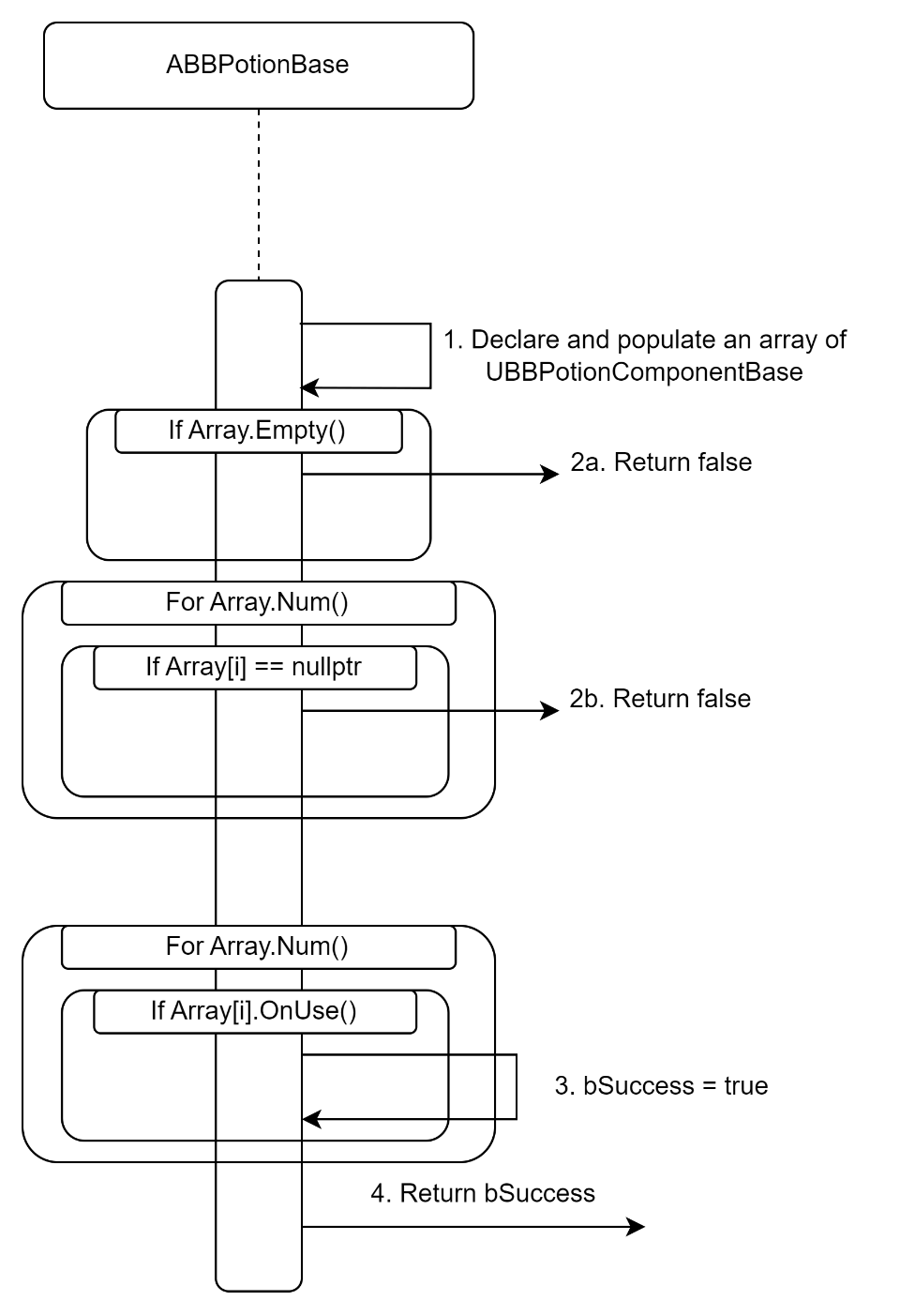
Description automatically generated with low confidence Table

Description automatically generated Graphical user interface, text

Description automatically generated

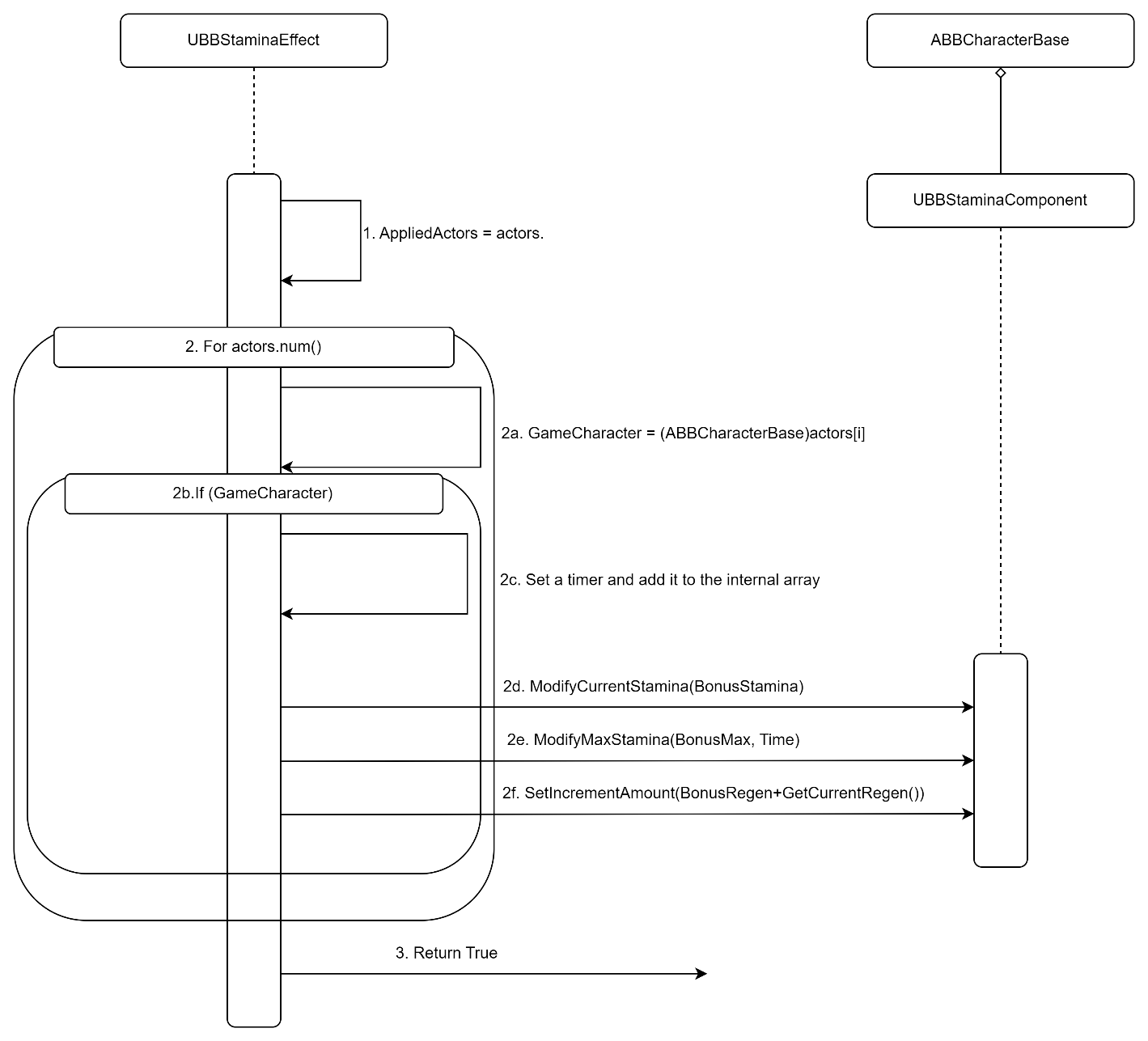
# Process View

## Calling OnUse on a BasePotion



1. Declaring and populating an TArray of UBBPotionComponentBase\*.
2. Error checking
   1. If the array is empty, return false.
   2. If the array has a component that’s set to nullptr, return false.
3. Loop through the components, calling OnUse on each of them. If any of them return true, set bSuccess to true.
4. Return bSuccess.

## Calling OnUse on a UBBStaminaEffect

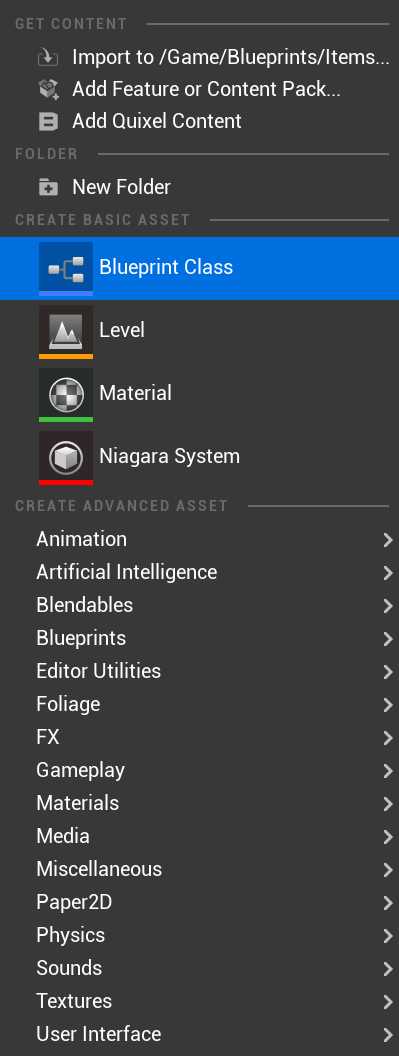
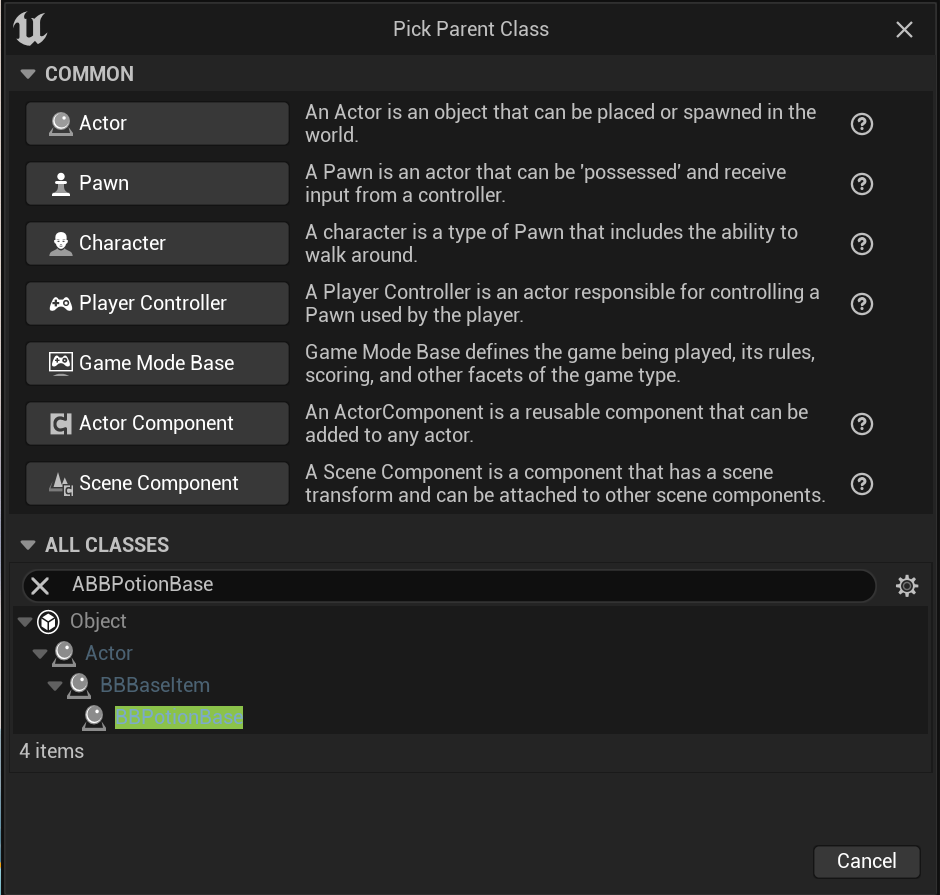


1. Copy the actors that were passed in to the member array AppliedActors.
2. For each actor;
   1. Cast the actor to a ABBCharacterBase.
   2. Make sure the Cast was successful (This makes sure the actor has a stamina comp we can effect).
   3. Setting a timer;
      1. Declaring an FTimerHandle.
      2. Setting the timer, passing in the FTimerHandle, this, the EndEffect function, and the Time.
      3. Add the handle to the internal array of FTimerHandles.
   4. Modify the StaminaComponents Current Stamina.
   5. Modify the StaminaComponents Max Stamina.
   6. Modify the StaminaComponents Stamina Regen.
3. Return true, the potion should never not be used.

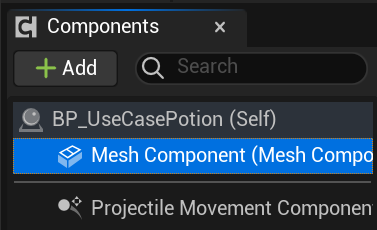
# Use Case View

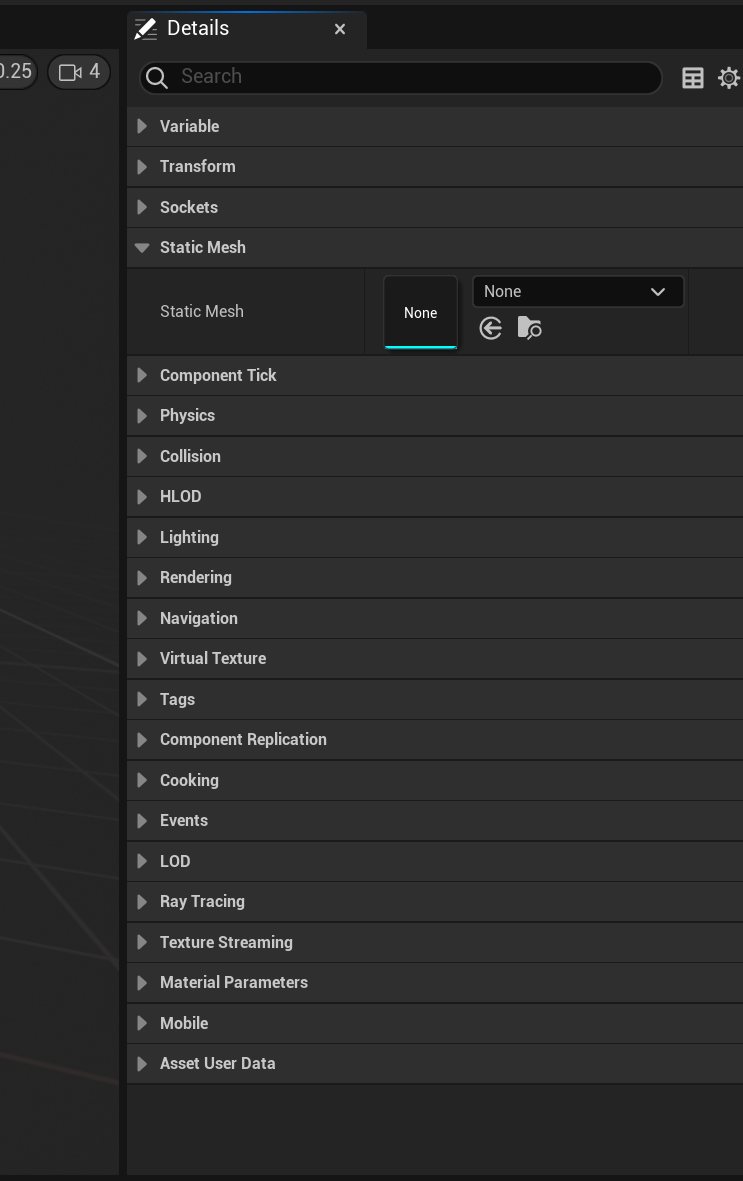
## Creating a custom potion

1. Create a Blueprint based on ABBPotionBase.

1. Set a mesh for the Mesh Component.





1. **Creating Effects:**
   1. Add a Component; In the blueprint editor, select the root, and under the Blightbrew tab, add the desired component.  
        
      ----------------------------------------------------------------------------------------------------------------  
      Graphical user interface

      Description automatically generated
   2. Once done, you’ll be able to expand that index, in there, you can add the desired effects.  
        
      ----------------------------------------------------------------------------------------------------------------  
      Graphical user interface, application

      Description automatically generated  
      -----------------------------------------------------------------------------------------------------------------  
      Graphical user interface

      Description automatically generated
   3. Once added, you can access the effect by opening up that index, and adjusting the effects to your liking.
   4. Repeat a – c for each way you want to use the potion.